```
!pip install advertools
!pip install adviz
!pip install searchconsole
!pip install dash_bootstrap_templates
import calendar
import advertools as adv
import adviz
import pandas as pd
import plotly.graph_objects as go
import plotly.express as px
import searchconsole
from dash_bootstrap_templates import load_figure_template
load_figure_template(['darkly', 'cosmo', 'bootstrap', 'flatly'])
```

```
query_df = pd.read_csv('/content/sample_data/country_query_month.csv')
page_df = pd.read_csv('/content/sample_data/page_per_month.csv')
```

```
# Data: monthly queries by country (clicks, impressions, ctr, position)
query_df = query_df[~query_df['country'].isin(['zzz', 'xkk'])].reset_index(drop=True)
query_df
```

```
# Monthly metrics by page and country
page_df
```

```
# Create subsets of the data
top_words = adv.word_frequency(query_df['query'], query_df['impressions'])
adviz.style_table(
    top_words.head(20),
    column_types=['text', 'text', 'bar', 'text'],
    column_widths=[0.15, 0.1, 0.4, 0.15],
    theme='flatly',
    width=900,
    height=600)
```

```
term_regex = [
    ('brand', 'advertool'),
    ('seo', 'seo'),
    ('python', 'python'),
    ('sitemap', 'sitemap|xml'),
    ('analysis', 'analy[sz]|analyt'),
    ('robots', 'robots'),
    ('crawl', 'crawl|scrap[ei]|spider'),
    ('log', '\blog(file)?'),
    ('search', 'search'),
    ('serp', 'serp'),
    ('google', 'google')
]
pd.DataFrame(term_regex, columns=['term', 'regex'])
```

```
for term, regex in term_regex:
    query_df[f'{term}_term'] = query_df['query'].str.contains(regex, regex=True)
query_df.filter(regex='^query').head().style.background_gradient()
```

```
# topics across all queries
(query_df
.filter(regex='_term')
.mean()
.sort_values(ascending=False)
.to_frame()
.rename(columns={0: '%'})
.style
.format('{:.1%}')
.bar(color='steelblue'))
```

```
query_df['serp_page'] = query_df['position'].div(10).astype(int).add(1)
query_df.sample(10)[['query', 'position', 'serp_page']].reset_index(drop=True)
```

```
month_country_ranks = pd.pivot_table(query_df, index=['date', 'country'], values=['clicks', 'impressions'], aggfunc='sum').reset_index
month_country_ranks['monthly_rank_clicks'] = month_country_ranks.groupby('date')['clicks'].rank(ascending=False)
month_country_ranks['flag'] = [adviz.flag(cc) for cc in month_country_ranks['country']]
monthly_impressions = pd.pivot_table(query_df, index=['date', 'country'], values='impressions', aggfunc='sum').reset_index()
monthly_impressions['flag'] = [adviz.flag(cc) for cc in monthly_impressions['country']]
top20_countries = pd.pivot_table(monthly_impressions, index='country', values='impressions', aggfunc='sum').sort_values('impressions',
# Country comparison (monthly impressions)
fig = px.line(
    monthly_impressions[monthly_impressions['country'].isin(top20_countries)],
    x='date',
    y='impressions',
    color='flag',
    template='flatly',
    height=700,
    hover_name='country')
fig.layout.legend.title.text = 'country'
for trace in fig.data:
    trace.visible = 'legendonly'
    trace.line.width = 4
fig.layout.legend.font.size = 30
fig.layout.legend.title.font.size = 15
fig.layout.hovermode = 'x unified'
fig.layout.hoverlabel.font.size = 15
fig.layout.xaxis.showgrid = False
fig


word_freq_dfs_bi = []

for month in query_df['date'].drop_duplicates():
    tempdf = query_df[query_df['date'].eq(month) ]
    query_metric = pd.pivot_table(tempdf, index='query', values='clicks', aggfunc='sum').reset_index()
    word_freq_df = adv.word_frequency(query_metric['query'], query_metric['clicks'], phrase_len=2).head(20)[['word', 'wtd_freq']]
    word_freq_df.insert(0, 'date', tempdf['date'].iloc[0])
    word_freq_dfs_bi.append(word_freq_df)
word_freq_clicks_bi = pd.concat(word_freq_dfs_bi)
n = 15
fig = adviz.racing_chart(
    word_freq_clicks_bi[['word', 'wtd_freq', 'date']],
    n=n,
    height=800,
    title=f'GSC Top {n} bigrams per month - clicks',
    theme='flatly')
fig.layout.yaxis.tickfont.size = 20
fig.layout.yaxis.title = 'bigram'
fig.layout.xaxis.title = 'weighted frequency'
fig


# Monthly Impressions (Map)
fig = px.choropleth(
    month_country_ranks,
    color='impressions',
    locations=month_country_ranks['country'].str.upper(),
    animation_frame='date',
    hover_name='flag',
    template='flatly',
    title='Google Search Console - Monthly Impressions by Country 2023',
    projection='natural earth',
    color_continuous_scale='blues',
    height=800)
fig.layout.geo.showframe = False
fig.layout.geo.lataxis.range = [-53, 76]
fig.layout.geo.lonaxis.range = [-137, 168]
fig.data[0].marker.line.color = 'gray'
fig


n = 15
fig = adviz.racing_chart(
    month_country_ranks[['flag', 'impressions', 'date']],
    n=n,
    height=800,
    # width=900,
    title=f'GSC Top {n} countries per month - impressions',
    theme='flatly')
fig.layout.yaxis.tickfont.size = 25
fig
```

```
n = 15
fig = adviz.racing_chart(
    month_country_ranks[['flag', 'clicks', 'date']],
    n=n,
    height=800,
    title=f'GSC Top {n} countries per month - clicks',
    theme='flatly')
fig.layout.yaxis.tickfont.size = 25
fig


# Monthly clicks (Map)
fig = px.choropleth(
    month_country_ranks,
    color='clicks',
    locations=month_country_ranks['country'].str.upper(),
    animation_frame='date',
    hover_name='flag',
    template='flatly',
    title='Google Search Console - Monthly Clicks by Country 2023',
    projection='natural earth',
    color_continuous_scale='blues',
    height=800)
fig.layout.geo.showframe = False
fig.layout.geo.lataxis.range = [-53, 76]
fig.layout.geo.lonaxis.range = [-137, 168]
fig.data[0].marker.line.color = 'gray'
fig


# Monthly impressions (clicks by country)
df = month_country_ranks[month_country_ranks['monthly_rank_clicks'].lt(21)]
fig = px.scatter(
    df,
    x='clicks',
    y='impressions',
    animation_frame='date',
    title='Monthly impressions ~ clicks by country (top 20)',
    text='flag',
    template='flatly',
    range_x=(0, df['clicks'].max() * 1.1),
    range_y=(0, df['impressions'].max() * 1.1),
    height=600
)
for frame in fig.frames:
    frame.data[0].textfont.size = 30
fig.frames[0].data[0].textfont.size = 30
for button in fig.layout.updatemenus[0].buttons:
    button.visible = False
fig


imp_clicks = pd.DataFrame({
    'impressions': query_df.groupby('date')['impressions'].sum(),
    'clicks': query_df.groupby('date')['clicks'].sum(),
}).reset_index().assign(ctr=lambda df: df['clicks'].div(df['impressions']))
imp_clicks['count'] = pd.pivot_table(query_df, index='date', values='impressions', aggfunc='count')['impressions'].rename('count').tol
url_dir_df = adv.url_to_df(page_df['page']).filter(regex='dir_|last_dir')
page_url_df = pd.concat([page_df, url_dir_df], axis=1)
monthly_page_imp = pd.pivot_table(page_url_df, index=['date','last_dir'], values='impressions', aggfunc='sum').reset_index()
monthly_page_imp['last_dir'] = monthly_page_imp['last_dir'].str.replace('\.html$', '', regex=True)


fig = adviz.racing_chart(
    monthly_page_imp[['last_dir', 'impressions', 'date']],
    n=20,
    title='Top 15 pages monthly (impressions)',
    height=850, theme='flatly')
fig.layout.margin.l = 200
fig.layout.yaxis.title = None
fig.layout.yaxis.tickfont.size = 16
fig
```

```
# Query count (monthly)
query_count = pd.pivot_table(
    query_df,
    index='date',
    values='query',
    aggfunc=pd.Series.nunique).reset_index()
fig = px.line(
    query_count,
    x='date',
    y='query',
    markers=True,
    template='flatly',
    height=600,
    title='Monthly query count')
fig.data[0].marker.size = 20
fig.layout.yaxis.title = 'count'

fig


# Correlation matrix (date, impressions, clicks)
fig = px.scatter_matrix(imp_clicks, height=800, template='flatly')
fig.update_traces(diagonal_visible=False, showupperhalf=False)
fig
```